ACPI for Arm Components 1.1

Platform Design Document

Non-confidential

Document number: DEN0093

Contents

Release information

Date	Version	Changes
2021/Nov/01	1.1	• External release • Added HID for the HD LCD available in some Arm development platforms such as the Juno. • Added clarifying text on the Arm Generic UART and its relation to the PL011. • Added support for CMN-700, CMN-650 and the CoreLink Networkon-chip Interconnect family. • Added support for DSU-110. • Added chapter to describe HIDs for Arm standards based features. • Added reference to Arm BSA specification, and updated language to reflect nomenclature thereof. • Updated HID definition for Arm CoreSight PMU Architecture PMU device to match definitions in DEN0117. • Added the ACPI AGDI table. This table describes a Generic Diagnostic Dump and Reset device interface. • Added clarifying text on the differences between CMN-600 and other CMN networks in terms of the way their register blocks are organized. • Updated device object description for NI-xy0 series to support description of more than one PMU overflow interrupts.
2020/Jul/12	1.0	• External release

Arm Non-Confidential Document Licence (“Licence”)

This Licence is a legal agreement between you and Arm Limited (“**Arm**”) for the use of Arm’s intellectual property (including, without limitation, any copyright) embodied in the document accompanying this Licence (“**Document**”). Arm licenses its intellectual property in the Document to you on condition that you agree to the terms of this Licence. By using or copying the Document you indicate that you agree to be bound by the terms of this Licence.

“**Subsidiary**” means any company the majority of whose voting shares is now or hereafter owner or controlled, directly or indirectly, by you. A company shall be a Subsidiary only for the period during which such control exists.

This Document is **NON-CONFIDENTIAL** and any use by you and your Subsidiaries (“Licensee”) is subject to the terms of this Licence between you and Arm.

Subject to the terms and conditions of this Licence, Arm hereby grants to Licensee under the intellectual property in the Document owned or controlled by Arm, a non-exclusive, non-transferable, non-sub-licensable, royalty-free, worldwide licence to:

(i) use and copy the Document for the purpose of designing and having designed products that comply with the Document;

(ii) manufacture and have manufactured products which have been created under the licence granted in (i) above; and

(iii) sell, supply and distribute products which have been created under the licence granted in (i) above.

Licensee hereby agrees that the licences granted above shall not extend to any portion or function of a product that is not itself compliant with part of the Document.

Except as expressly licensed above, Licensee acquires no right, title or interest in any Arm technology or any intellectual property embodied therein.

THE DOCUMENT IS PROVIDED “AS IS”. ARM PROVIDES NO REPRESENTATIONS AND NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, INCLUDING, WITHOUT LIMITATION, THE IMPLIED WARRANTIES OF MERCHANTABILITY, SATISFACTORY QUALITY, NON-INFRINGEMENT OR FITNESS FOR A PARTICULAR PURPOSE WITH RESPECT TO THE DOCUMENT. Arm may make changes to the Document at any time and without notice. For the avoidance of doubt, Arm makes no representation with respect to, and has undertaken no analysis to identify or understand the scope and content of, third party patents, copyrights, trade secrets, or other rights.

NOTWITHSTANING ANYTHING TO THE CONTRARY CONTAINED IN THIS LICENCE, TO THE FULLEST EXTENT PETMITTED BY LAW, IN NO EVENT WILL ARM BE LIABLE FOR ANY DAMAGES, IN CONTRACT, TORT OR OTHERWISE, IN CONNECTION WITH THE SUBJECT MATTER OF THIS LICENCE (INCLUDING WITHOUT LIMITATION) (I) LICENSEE’S USE OF THE DOCUMENT; AND (II) THE IMPLEMENTATION OF THE DOCUMENT IN ANY PRODUCT CREATED BY LICENSEE UNDER THIS LICENCE). THE EXISTENCE OF MORE THAN ONE CLAIM OR SUIT WILL NOT ENLARGE OR EXTEND THE LIMIT. LICENSEE RELEASES ARM FROM ALL OBLIGATIONS, LIABILITY, CLAIMS OR DEMANDS IN EXCESS OF THIS LIMITATION.

This Licence shall remain in force until terminated by Licensee or by Arm. Without prejudice to any of its other rights, if Licensee is in breach of any of the terms and conditions of this Licence then Arm may terminate this Licence immediately upon giving written notice to Licensee. Licensee may terminate this Licence at any time. Upon termination of this Licence by Licensee or by Arm, Licensee shall stop using the Document and destroy all copies of the Document in its possession. Upon termination of this Licence, all terms shall survive except for the licence grants.

Any breach of this Licence by a Subsidiary shall entitle Arm to terminate this Licence as if you were the party in breach. Any termination of this Licence shall be effective in respect of all Subsidiaries. Any rights granted to any Subsidiary hereunder shall automatically terminate upon such Subsidiary ceasing to be a Subsidiary.

About this document

Terms and abbreviations

Term	Meaning
ACPI	Advanced Configuration and Power Interface
ASL	ACPI Source Language
CMN	Arm CoreLink Mesh Network
DSU	DynamicIQ Shared Unit
DTC	Debug and Trace Controller
GIC	Arm Generic Interrupt Controller
GSIV	Global System Interrupt Vector
HN	Home Node
PMU	Performance Monitoring Unit
RN	Requester Node
SDEI	Software Delegated Exception Interface
SPE	Statistical Profiling Extension
XP	Cross-point

References

This section lists publications by Arm and by third parties.

See Arm Developer (http://developer.arm.com) for access to Arm documentation.

[1] *Advanced Configuration and Power Interface Specification*. UEFI Forum, https://uefi.org/specifications.

[2] *Arm® Architecture Reference Manual Armv8, for Armv8-A architecture profile: ARM DDI 0487F.b (ID040120)*. Arm Limited.

[3] *Arm® Generic Interrupt Controller Architecture Specification, GIC architecture version 3 and version 4: Arm IHI 0069E (ID012119)*. Arm Limited.

[4] *Arm® System Memory Management Unit Architecture Specification SMMU architecture versions 3.0, 3.1 and 3.2: Arm IHI 0070C.a*..

[5] *Arm® CoreSight™ Architecture Specification v3.0: Arm IHI 0029E (ID022717)*. Arm Limited.

[6] *Arm® Base System Architecture 1.0: DEN0094*. Arm Limited.

[7] *Arm IO Remapping Table: DEN0049E*. Arm Limited.

[8] *ACPI for CoreSight™ 1.1: DEN0067*. Arm Limitedd.

[9] *Arm® Functional Fixed Hardware Specification Document number: Arm DEN 0048A*. Arm Limited, https://uefi.org/acpi.

[10] *_DSD (Device Specific Data) Implementation Guide v1.2*. UEFI Forum, https://uefi.org/specifications.

[11] *ACPI for Arm v8-A Memory System Resource Partitioning and Monitoring: DEN0065*. Arm Limited.

[12] *Arm Architecture Reference Manual Supplement Memory System Resource Partitioning and Monitoring (MPAM), for Armv8-A: ARM DDI 0598B.a*. Arm Limited.

[13] *ACPI for the Armv8-A RAS Extensions 1.0: DEN0085*. Arm Limited.

[14] *Arm® Reliability, Availability, and Serviceability (RAS) Specification: Arm DDI 0587C.b*. Arm Limited.

[15] *ACPI for Arm® CoreSight™ Performance Monitoring Unit Architecture: DEN0117*. Arm Limited.

[16] *Arm® CoreSight™ Performance Monitoring Unit Architecture Specification*. Arm Limited.

[17] *Serial Port Console Redirection Table*. Microsoft Corporation, https://uefi.org/acpi.

[18] *Debug Port Table 2*. Microsoft Corporation, https://uefi.org/acpi.

[19] *Arm® DynamIQ™ Shared Unit, Revision r0p2, Technical Reference Manual: Arm 100453_0002_00_en*. Arm Limited.

[20] *https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-coherent-mesh-network-family*. Arm Limited.

[21] *https://developer.arm.com/ip-products/system-ip/corelink-interconnect/corelink-network-interconnect-family*. Arm Limited.

[22] *Arm® Power State Coordination Interface Platform Design Document: DEN0022D*. Arm Limited.

[23] *Software Delegated Exception Interface: DEN0054*. Arm Limited.

Feedback

Arm welcomes feedback on its documentation.

If you have comments on the content of this manual, send an e-mail to errata@arm.com. Give:

- The title (ACPI for Arm Components).
- The document ID and version (DEN0093 1.1).
- The page numbers to which your comments apply.
- A concise explanation of your comments.

Arm also welcomes general suggestions for additions and improvements.

1 Introduction

This document provides guidance for describing system components implemented or licensed by Arm, and their properties, in ACPI [1].

This specification does not cover ACPI description of the Arm Architecture [2] or related architectures, for example the Generic Interrupt Controller Architecture [3], the System Memory Management Unit Architecture [4], the CoreSight Architecture [5], or architected components that are described in the Base System Architecture [6]. The ACPI descriptions of these architectural components are covered in the following specifications instead:

Table 3: Arm Architectures that are covered by ACPI Tables

Specification	*Table or object*	*Arm Architecture Covered*
ACPI [1]	MADT	Arm Architecture [2]
ACPI [1]	MADT	GIC [3], SPE
I/O Remapping Table [7]	IORT	SMMU [4]
ACPI For CoreSight [8]	ACPI graph	CoreSight Architecture [5]
FFH for Arm [9]	FFH	Armv8 Activity Monitors Extension
ACPI _DSD Implementation Guide [10]	ACPI graph	CoreSight Architecture [5]
ACPI for MPAM [11]	MPAM	Armv8 MPAM Architecture [12]
ACPI for Arm RAS Extensions [13]	AEST	Armv8 RAS Extensions Architecture [14]
ACPI for Arm CoreSight PMU Architecture [15]	APMT	Armv8 CoreSight PMU Architecture [16]

The following table details coverage of specific Arm components in ACPI:

Table 4: Arm Components that are covered by specific ACPI Tables

ACPI Tables	*Arm Components Covered*
GTDT	Arm Generic Watchdog timer [6]
PPTT	Caches, cores, clusters
SPCR [17], DBG2 [18],	Arm Generic UART, PL011

This specification recommends that components that are not covered in standard or Arm-specific ACPI tables are described as ACPI device objects in ASL. Because a component might have multiple internal interfaces, Arm recommends that a separate ACPI device object is created to cover each of these interfaces to allow device drivers in OS to unequivocally bind to those specific interfaces. Interfaces like RAS Extensions and MPAM are not covered, and instead are described in appropriate tables, as indicated in Table 3 above.

The device objects should define the generic and specific properties of the component interface to aid software discovery from the OS, where:

- Generic properties include an ACPI namespace identifier for the component or its interface, the memory-mapped base address of the component or its interface, and the interrupts that the component can generate.
- Specific properties are component specific or vendor specific or both.

2 ACPI for Arm Components

2.1 ACPI Identifiers

ACPI Identifiers of Arm components follow the conventions that are described in [1]. The ACPI Hardware ID object, _HID, is used as the primary identifier. For Arm components, the format is ARMH####.

For some Arm components, existing standard ACPI or PNP identifiers may also be used as _HID values. In such cases, Arm recommends setting the _CID of the device object as ARMH#### where relevant.

2.2 Reserved ACPI IDs for legacy Arm components

This section lists reserved ACPI IDs for components from Arm that are managed as a complete unit by a single device driver, or are required for cross-referencing from other Arm ACPI tables like the AEST [13] and the MPAM [12]. These components are listed in Table 5.

Table 5: Reserved ACPI IDs for legacy Arm components

Component	HID
Prime cell UART (PL011)	ARMH0011
Prime cell General Purpose I/O	ARMH0061
Arm HD LCD	ARMH0002

2.3 Reserved ACPI IDs for Arm components defined by Arm BSA

The following types of Arm components require a unique ACPI ID for identification:

- Specifically defined by the BSA specification [6]
- Not described in any standard ACPI table listed in Table 3 or Table 4
- Must be described in ACPI namespace by virtue of the above properties

Table 6: Reserved ACPI IDs for BSA components

BSA Component	HID	Notes
Arm Generic UART	ARMHB000	Some operating systems use the PL011 HID (see Table 5 above) to bind to the Arm Generic UART in the system. While this practice is flawed and not encouraged by Arm, Arm acknowledges that it must be permitted until formal support for the Arm Generic UART HID is made available in these operating systems. Arm strongly recommends use of the Arm Generic UART HID going forward.

A BSA component can be described in both ACPI namespace and the standard ACPI tables in a given system. For example, a system might have two instances of the Arm Generic UART: a primary UART that is used by the OS and a secondary, general-purpose, UART for application usage. In this example system, the primary

UART must be declared in the ACPI SPCR table. The secondary UART might be declared as a device object in ACPI namespace and assigned the HID that is defined in Table 6.

2.4 Reserved ACPI IDs for features based on Arm specifications

This section lists reserved ACPI IDs for components that are based on features described in Arm specifications. Such components are primarily described in Arm-specific ACPI tables, but might also have additional vendor-specific properties that can only be described by means of ACPI device objects in DSDT. The recommended practice is to link such device objects to the corresponding table node in the Arm ACPI table that describes the component. The standard HIDs for these device objects enable the OS to understand that they can be managed by standard drivers that conform to the Arm ACPI table based enumeration and configuration of the components.

Table 7: HIDs for features described in Arm specifications

Component	HID
MPAM Memory System Component (MSC)	ARMHAA5C
CoreSight PMU Architecture PMU	ARMHE001

2.5 Reserved ACPI IDs for generic devices

This section describes devices that are generic across Arm implementations and are not tied to an IP implementation.

2.5.1 Generic Diagnostic Dump and Reset Device Interface

Some use-cases, such as system management, require the ability to generate a non-maskable event to the OS to request the OS kernel to perform a diagnostic dump and reset the system. This section describes the ACPI representation for a Generic Diagnostic Dump and Reset Device for Arm systems that serves as an interface for initiating the non-maskable event on behalf of requesting agents.

The Generic Diagnostic Dump and Reset device is described using the ACPI table representation outlined in Section 2.5.1.1.

The interface supports both SDEI and native interrupt based generation of non-maskable events.

2.5.1.1 ACPI Table Representation

Table 8: ACPI AGDI Table

Field	Byte length	Byte offset	Description
Header			Standard ACPI format for header.
Signature	4	0	‘AGDI’, Arm Generic Diagnostic Dump and Reset Interface table.
Length	4	4	Length of this table in bytes.
Revision	1	8	Must be 0 for version 1.0 of this specification.
Checksum	1	9	The entire table must sum to zero.
OEM ID	6	10	OEM ID.

Field	Byte length	Byte offset	Description
OEM Table ID	8	16	The table ID is the manufacture model ID.
OEM Revision	4	24	OEM revision of the AGDI table for the supplied OEM Table ID.
Creator ID	4	28	The vendor ID of the utility that created the table.
Creator Revision	4	32	The revision of the utility that created the table.
Body			
Flags node structures	1	36	Bit 0: Signaling mode: 0b: SDEI-based Signaling mode. 1b: Interrupt-based Signaling mode. Other values are reserved for future use by this specification and must be zero for this version of the specification.
Reserved	3	37	Reserved, must be zero.
SDEI Event number	4	40	SDEI Event number of the event generated by the device. This field is valid only if the signaling mode is set to 0b.
GSIV	4	44	The GSIV of the interrupt that is generated by the device. This field is valid only if the signaling mode is set to 1b.

2.6 Arm components requiring ACPI description

This section describes Arm components that have at least one interface that is not covered by standard or Arm-specific ACPI tables, and that must be described in ACPI namespace.

2.6.1 Arm DMC620 Memory Controller

Table 9 describes interfaces within the DMC620 that require ACPI description to support software discovery.

2.6.1.1 Interface identification

Table 9: Arm DMC620 Memory Controller HID values

Value	Description
ARMHD620	ACPI Hardware Identifier for the DMC620 PMU.

2.6.1.2 The DMC620 PMU

The DMC620 PMU is assigned the HID value of ARMHD620, as specified in Table 9.

Device configuration objects for the DMC620 PMU

Table 10: Configuration objects for the DMC620 PMU

Object	Values	Type	Description
_CRS	Base address	QWordMemory	Base address of the PMU in the system address map

Object	Values	Type	Description
	GSIV	Interrupt	GSIV of the PMU overflow interrupt

ASL reference code for the DMC620 PMU

The DMC620 PMU register space is mapped at a 512-byte range that begins at offset 0x80000A00 in the system address space. In this reference code, it is assumed that the PMU overflow interrupt from the DMC620 is mapped to GSIV 312.

```
Device(MC00) { // PMU interface on the example DMC620 memory controller
               // instance in the system.

       Name(_HID, "ARMHD620")
       Name(_CID, "ARMHD620")
       Name(_UID, 0)
       Name(_CCA, 1)
       Name(_STR, Unicode("Socket0:MCU0"))
       Name(_STA, 0, NotSerialized) {
          Return (0x0f)
       }

       Name(_CRS, ResourceTemplate () {
                // Descriptor for 64-bit memory-mapped register space
                // of the DMC620
                QWordMemory (
                   ResourceProducer,     // ResouceUsage
                   PosDecode,            // Decode
                   MinFixed,             // Min range is fixed
                   MaxFixed,             // Max range is fixed
                   NonCacheable,         // Cacheable
                   ReadWrite,            // ReadAndWrite
                   0x0000000000000000,   // Address Granularity - GRA
                   0x0000100080000A00,   // AddressMinimum - MIN
                   0x0000100080000BFF,   // AddressMaximum - MAX
                   0x0000000000000000    // AddressTranslation - TRA
                   0x0000000000000200,   // RangeLength - LEN
                   ,                     // ResourceSourceIndex
                   ,                     // ResourceSource
                   CFGS                  // DescriptorName
                )

                // PMU overflow Interrupt, with GSIV = 312
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {312}
       })
}
```

2.6.2 Arm DynamIQ Shared Unit (DSU)

The Arm DynamIQ Shared Unit (DSU) provides circuitry, logic, interfaces, and an optional shared cache to support a DynamIQ cluster. The DSU is described in detail in [19]. Interfaces within the DSU that require ACPI description to support software discovery are described here.

The DSU is shared by a set of cores organized as a cluster. The ACPI description of interfaces within the DSU is therefore expressed as a device object that is a child of the ACPI processor container object that describes the cluster.

The OS must parse the CPU topology in ACPI namespace to discover the DSU interface objects and to understand the associativity between those DSU instances and cores.

2.6.2.1 Interface Identification

Table 11: Arm DSU HID values

Value	Description
ARMHD500	ACPI Hardware Identifier for the DSU PMU.
ARMHD501	ACPI Hardware Identifier for the common DSU elements.
ARMHD510	ACPI Hardware Identifier for the DSU 110 PMU.

2.6.2.2 Common DSU elements

Common DSU elements are collectively described by a single device object with a HID of ARMHD501. This device object allows cross-referencing the DSU object from other ACPI objects and tables.

2.6.2.3 DSU PMU

The DSU provides a PMU for monitoring and recording miscellaneous events. DSU PMU registers are presented as System Registers to allow for native software discovery. Therefore, no ACPI description is required to locate them. When a PMU counter overflows, the PMU asserts an interrupt signal that can be routed to an interrupt controller such as the GIC.

The DSU PMU is assigned the HID value of ARMHD500, as specified in Table 11.

DSU PMU device configuration objects

Table 12: Arm DSU PMU device configuration objects

Object	Value	Type	Description
_CRS	GSIV	Interrupt	GSIV of the DSU PMU overflow interrupt

ASL reference code

This reference code illustrates how the CPU topology that is associated with the DSU is described in ACPI namspace by including the DSU device objects in the CPU hierarchy description. The code showcases an example system that has two clusters, each with a single DSU. Each cluster includes two CPU cores that share the associated DSU.

The code also illustrates how the global DSU device object may be also included within the CPU topology description. The placement of the global DSU object allows generic references to the DSU from the OSPM.

```
Device (SYSM)  { // System level states
       Name (_HID, "ACPI0010")
       Name (_UID, 0)
       Name (_LPI,
          Package() {...}
       )

       Device (CLU0) { // Cluster 0
          Name (_HID, "ACPI0010")
          Name (_UID, 1)
```

```
    Name (_LPI,
       Package() {...}
    )

    Device (DSU0) { // Common DSU Instance 0 associated with cluster 0
       Name (_HID, "ARMHD501")
       Name (_UID, 0)
    } // DSU descriptor ends here

    Device (DSP0) { // PMU interface on DSU 0 associated with cluster 0
       Name (_HID, "ARMHD500")
       Name (_UID, 0)
       Name(_CRS, ResourceTemplate () {
        // PMU overflow Interrupt, with GSIV = 302
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {302}
       })
    } // DSU0 PMU interface descriptor ends here

    Device (CPU0) { // Core0
       Name (_HID, "ACPI0007")
       Name (_UID, 0)
       Method (_LPI, 0, NotSerialized) {
          ...
       }
    } // CPU0 description ends here

    Device (CPU1) { // Core1
       Name (_HID, "ACPI0007")
       Name (_UID, 1)
       Method (_LPI, 0, NotSerialized) {
          ...
       }
    } // CPU1 description ends here
} // Cluster 0 descriptor ends here

Device (CLU1) { // Cluster 1
    Name (_HID, "ACPI0010")
    Name (_UID, 2)
    Method (_LPI, 0, NotSerialized) {
       ...
    }

    Device (DSU1) { // Common DSU Instance 1 associated with cluster 1
       Name (_HID, "ARMHD501")
       Name (_UID, 1)
    } // DSU descriptor ends here

    Device (DSP1) { // PMU interface on DSU 1 associated with cluster 1
       Name (_HID, "ARMHD500")
       Name (_UID, 1)
       Name(_CRS, ResourceTemplate () {
        // PMU overflow Interrupt, with GSIV = 402
        Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {402}
       })
    } // DSU1 PMU interface descriptor ends here

    Device (CPU2) { // Core2
       Name (_HID, "ACPI0007")
       Name (_UID, 2)
       Method (_LPI, 0, NotSerialized) {
```

```
            ...
          }
        } // CPU2 description ends here

        Device (CPU3) { // Core3
          Name (_HID, "ACPI0007")
          Name (_UID, 3)
          Method (_LPI, 0, NotSerialized) {
            ...
          }
        } // CPU3 description ends here
    } // Cluster 1 descriptor ends here
} // End of system description
```

2.6.3 Arm CoreLink CMN Coherent Mesh Network Family

The Arm CoreLink CMN-xy0 Coherent Mesh Network components are described in [20]. The CMN-xy0 network consists of Cross-points (XPs), Home Nodes (HNs) and Request Nodes (RNs). A special Home Node, HN-D, houses the global configuration registers. In this network, PMU logic is integrated into Debug and Trace Logic Controllers (DTCs). HN-D also houses the primary DTC, which is labelled DTC0. Other DTCs, DTC1 to DTC3, are housed in special Home Nodes called HN-T. The CMN-xy0 layout is illustrated in the following diagram:

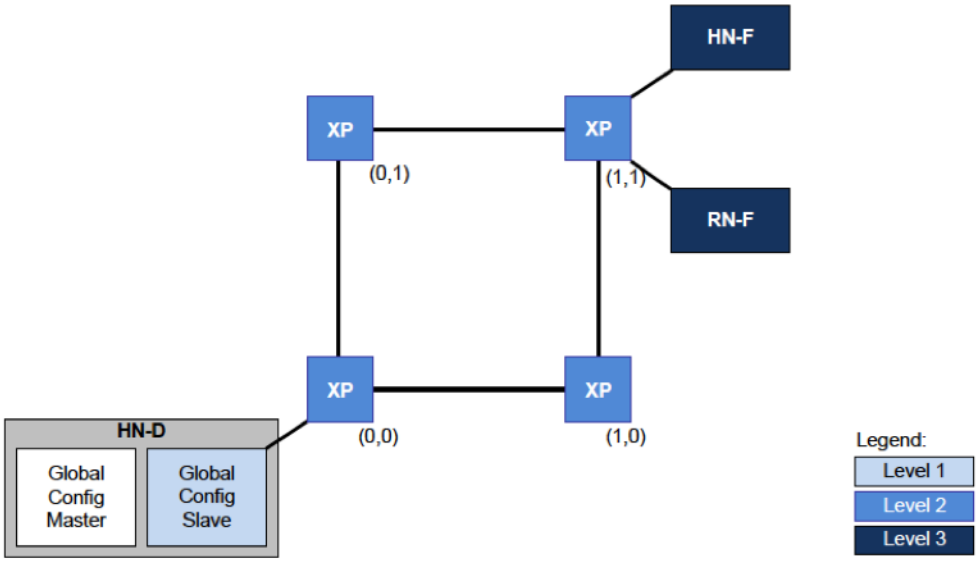

Figure 1: High-level layout of the CMN-xy0

All configuration registers that belong to the CMN-xy0 network are mapped into System Address Map (SAM) at a pre-determined, 64MB aligned address range at offset PERIPHBASE. For the CMN-600, the configuration space of the root node of the network is mapped to an address range at offset ROOTNODEBASE. These offsets are illustrated in the following diagram:

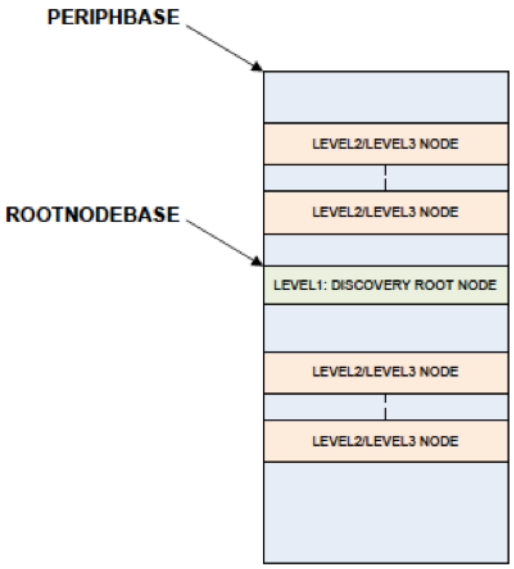

Figure 2: Memory-mapped configuration region of the CMN-600

All registers are organized into one or more register blocks. Each register block is 16KB in size, and there is one register block for each logical block in the network. Each register block is called a node. The nodes are laid out in a tree hierarchy. For the CMN-600, the root of the tree hierarchy is the root node, as illustrated in the preceding diagram. For the other CMN series products, the register blocks start at PERIPHBASE itself.

2.6.3.1 Interface Identification

Notation: The HID is composed of the following sub-identifier fields:

- The first four characters, “ARMH”, indicate that this is an Arm-specific hardware.
- The fifth character is set to ‘C’ to indicate that this is a product from the CoreLink interconnect family.
- The last three characters are set to the product number.

Table 13: Arm CMN-xy0 HID values

Value	Description
ARMHC600	ACPI Hardware Identifier for the CMN-600 PMU
ARMHC650	ACPI Hardware Identifier for the CMN-650 PMU
ARMHC700	ACPI Hardware Identifier for the CMN-700 PMU

2.6.3.2 Common CMN-xy0 elements

Common CMN-xy0 elements other than the PMU might be optionally described with a HID of ARMHCxy1. The common device object facilitates cross-referencing the CMN-xy0 object from other ACPI objects and tables.

2.6.3.3 CMN-xy0 PMU

CMN-xy0 PMU logic is integrated into Debug and Trace Logic Controllers (DTCs). Thus, discovery of the PMU interface in a CMN-xy0 network relies on the topology in which the DTCs are organized within the network.

The HN-D houses the primary DTC, which is labelled DTC0. Each DTC is associated with multiple DTMs. The parent DTC and its children DTMs form a DTC domain. The following diagram shows an example system with two DTC domains:

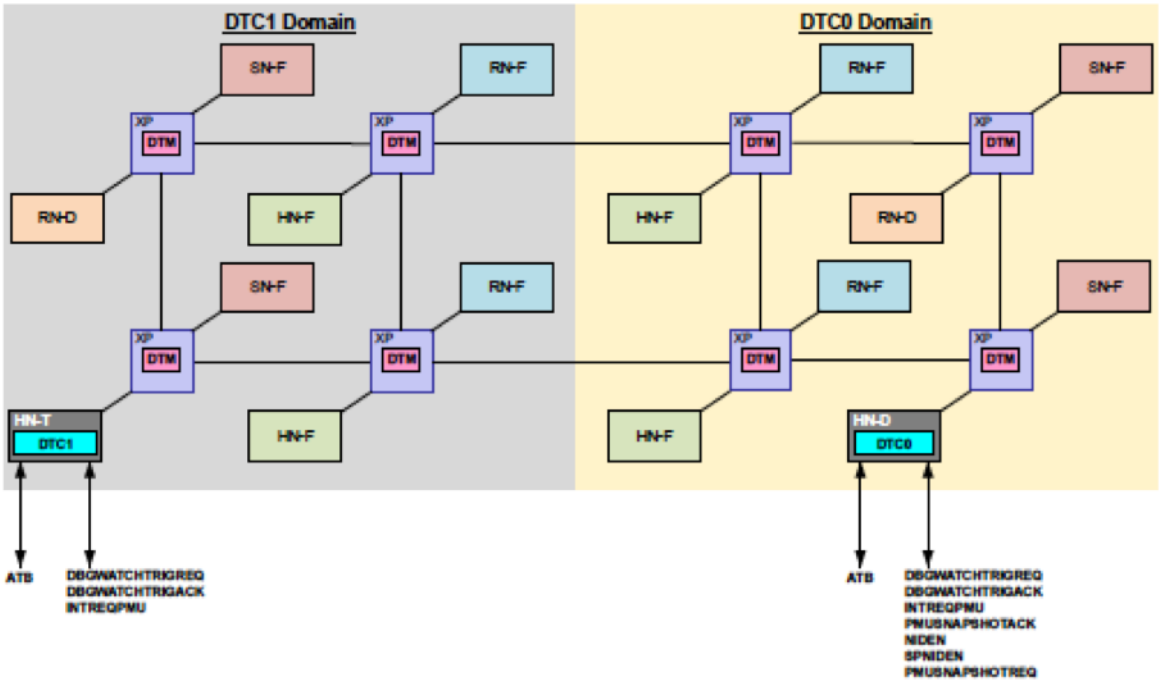

Figure 3: CMN-xy0-based topology with 2 DTCs

Each PMU asserts its own dedicated interrupt signal on overflow, called INTREQPMU. The interrupts from the PMUs are routed to the GIC, and appear as one or more GSIVs to software.

This means that, to support software discovery of CMN PMUs, the primary sources of required information are:

Table 14: Information sources for CMN-xy0 PMU discovery

Information	Information source
Discovery of PMUs in CMN-xy0 network. This is based on discovery of DTC*n* for PMU*n*	PERIPHBASE
Discovery of PMUs in CMN-600 only. For the CMN-600, register blocks are organized in a tree hierarchy that begins at offset ROOTNODEBASE.	ROOTNODEBASE
Identity of PMU overflow interrupt for all PMU logic within the network , where PMU*n* is housed in DTC*n*	INTREQPMU*n* routing

PMU overflow interrupt description ordering and Logical ID

The CMN-xy0 network specification allows for up to four DTCs in the system. DTC0 is the primary DTC, and must always exist. Additional DTCs may be installed on various cross-points, based on system design

and topology. The CMN-xy0 provides logical numbering of DTCs to allow for their unique identification in hardware. The Logical ID is the only way for software to identify a unique DTC. However, the Logical IDs is not guaranteed to be the same as the DTC number. For example, DTC 1 is not guaranteed to obtain a Logical ID of 1.

Here is an example mapping for a CMN-xy0 system with 4 DTCs is as follows:

Table 15: DTC domain number to Logical ID mapping in an example system

DTC Name	DTC Domain Number (Value)	DTC Logical ID (Notation)	DTC Logical ID (Value)	Description
DTC0	0	DTC[0]	0	DTC0 is always assigned Logical ID of 0.
DTC1	1	DTC[1]	*j*	DTC1 is assigned Logical ID *j*, where *j* != 0.
DTC2	2	DTC[2]	*k*	DTC2 is assigned Logical ID *k*, where *j* != 0 && *k* != *j*.
DTC3	3	DTC[3]	*m*	DTC1 is assigned Logical ID *m*, where *m* != 0 && *m* != *j* != *k*.

To address the lack of an explicit relation between the Logic ID and Domain Number as shown in Table 15, the PMU overflow interrupt descriptors are organized in increasing order of the hardware assigned Logical IDs of the related DTCs, respectively. This allows OS software to map interrupt descriptors to their parent DTCs. This means that INTDESC[1] corresponds to DTC0, INTDESC[2] corresponds to the DTC that is assigned the next smallest Logical ID, and so on.

Device Identification for CMN-xy0 PMU

The CMN-xy0 PMU interface is assigned an ACPI ID of ARMHCxy0 as indicated in Table 13.

Device configuration objects

Table 16: Arm CMN-xy0 Coherent Mesh Network configuration objects

Object	Values	Type	Description
_CRS	PERIPHBASE	QWordMemory	Base address of the memory-mapped region in the system address map where the CMN-xy0 registers are mapped.
	ROOTNODEBASE	QWordMemory	Base address of the root node. This field is specific to the CMN-600 device object.
	GSIV[*n*]	Interrupt	List of GSIVs of *n* distinct interrupts that can be generated by the *n* PMUs located in this instance of the CMN-xy0.

The list must always begin with PERIPHBASE, followed by ROOTNODEBASE if this is a CMN-600 device object, and finally the interrupt resource descriptors. Similarly, the interrupt resource descriptors for the PMUs,for example GSIV[*n*], must appear in numerically increasing order of the corresponding DTC[*n*]. So the first interrupt resource descriptor relates to DTC[0], the second interrupt resource descriptor relates to DTC[1] and so on. The *m*th interrupt resource descriptor in the list corresponds to DTC[*m*-1].

ASL code example

This code example showcases a CMN-600 based network with a dimension larger than 4x4 and that has two DTCs. Like the example illustrated in Figure 3 in Section 2.6.3.3, the first DTC is always DTC0 and is attached to HN-D. However, the second DTC could be assigned any Logical ID that will be non-zero. The term DTC[*n*] is used to represent the Logical ID of DTC*n*. In this example, PERIPHBASE of the CMN-600 is set as 0xAFE0000000, and the root node is at an offset of 0xC000 from PERIPHBASE.

```
Device(CMN6) { // CMN-600 device object for an X * Y mesh where X, Y > 4
       Name(_HID, "ARMHC600")
       Name(_CRS, ResourceTemplate () {
                // Descriptor for 256 MB of the CFG region at offset PERIPHBASE
                QWordMemory (
                   ResourceConsumer,     // bit 0 of general flags is 0
                   PosDecode,
                   MinFixed,             // Range is fixed
                   MaxFixed,             // Range is Fixed
                   NonCacheable,
                   ReadWrite,
                   0x00000000,           // Granularity
                   0xAFE0000000,         // Min, set to PERIPHBASE
                   0xAFEFFFFFFF,         // Max
                   0x00000000,           // Translation
                   0x0010000000,         // Range Length = 256MB
                   ,                     // ResourceSourceIndex
                   ,                     // ResourceSource
                   CFGR                  // DescriptorName
                )

                // Descriptor for the root node. This is a 16KB region at
                // offset ROOTNODEBASE. In this example, ROOTNODEBASE starts
                // at the 16KB aligned offset of PEIPHBASE + 0xC000
                QWordMemory (
                   ResourceConsumer,     // bit 0 of general flags is 0
                   PosDecode,
                   MinFixed,             // Range is fixed
                   MaxFixed,             // Range is Fixed
                   NonCacheable,
                   ReadWrite,
                   0x00000000,           // Granularity
                   0xAFE000C000,         // Min, set to ROOTNODEBASE
                   0xAFE000FFFF,         // Max
                   0x00000000,           // Translation
                   0x0000004000,         // Range Length = 16KB
                   ,                     // ResourceSourceIndex
                   ,                     // ResourceSource
                   ROOT                  // DescriptorName
                )

                // Interrupt on PMU0 overflow, attached to DTC[0], with GSIV = <
                    ↪gsiv0>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv0>}

                // Interrupt on PMU1 overflow, attached to DTC[1], with GSIV = <
                    ↪gsiv1>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv1>}
       })
}
```

The following ASL code example shows a CMN-700 based network that has a dimension less than 4x4 and that has four DTCs, DTC0-DTC3. The logical numbers of these DTCs are DTC[0] to DTC[3].

```
Device(CMN7) { // CMN-700 device object for an X * Y mesh where X, Y <= 4
       Name(_HID, "ARMHC700")
       Name(_CRS, ResourceTemplate () {
                // Descriptor for 256 MB of the CFG region at offset PERIPHBASE
                QWordMemory (
                   ResourceConsumer,     // bit 0 of general flags is 0
                   PosDecode,
                   MinFixed,             // Range is fixed
                   MaxFixed,             // Range is Fixed
                   NonCacheable,
                   ReadWrite,
                   0x00000000,           // Granularity
                   0xAFE0000000,         // Min
                   0xAFE0FFFFFF          // Max
                   0x00000000,           // Translation
                   0x0010000000,         // Range Length
                   ,                     // ResourceSourceIndex
                   ,                     // ResourceSource
                   CFGR                  // DescriptorName
                )

                // Interrupt on PMU0 overflow, attached to DTC[0], with GSIV = <
                    ↪gsiv0>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv0>}

                // Interrupt on PMU1 overflow, attached to DTC[1], with GSIV = <
                    ↪gsiv1>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv1>}

                // Interrupt on PMU2 overflow, attached to DTC[2], with GSIV = <
                    ↪gsiv2>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv2>}

                // Interrupt on PMU3 overflow, attached to DTC[3], with GSIV = <
                    ↪gsiv3>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv3>}
       })
}
```

2.6.4 Arm CoreLink Network-on-chip Interconnect Family

The Arm CoreLink NI-xy0 Coherent Network-on-chip Interconnect family is described in [21]. Configuration registers of the NI-x00 components are mapped into a dedicated memory-mapped address range at offset PERIPHBASE. The PMU registers are then available at an offset from PERIPHBASE in a 4K register block called a node. Overflow interrupts from each of the eight counters are wired together to a summary overflow interrupt.

2.6.4.1 Interface Identification

Notation: The HID is composed of the following sub-identifier fields:

- The first four characters, “ARMH”, indicate that this is an Arm-specific hardware.

- The fifth character is set to ‘C’ to indicate that this is a product from the CoreLink Network-on-chip interconnect family.
- The sixth character is set to ‘B’ to indicate that this is a product from the Arm CoreLink Network Interconnect family.
- The last two characters are set to the first two numbers of the product name.

Table 17: Arm NI-xy0 HID values

Value	Description
ARMHCB70	ACPI Hardware Identifier for the NI-700 PMU

Device configuration objects

Table 18: Arm NI-xy0 configuration objects

Object	Values	Type	Description
_CRS	PERIPHBASE	QWordMemory	Base address of the memory-mapped region in the system address map where the NI-xy0 registers are mapped
	GSIV[n]	Interrupt	List of GSIVs for the overflow interrupts of the PMU instances in the NI-xy0. GSIV[i] is associated with the *i*th PMU block.

ASL reference code example

The following ASL code example shows an NI-700 interconnect being described in DSDT. The NI-700 registers are mapped to PERIPHBASE of 0x2000000000 in this example.

```
Device(CNI0) { // NI-700 device object
       Name(_HID, "ARMHCB70")
       Name(_CRS, ResourceTemplate () {
                // Descriptor for PERIPHBASE
                QWordMemory (
                   ResourceConsumer,     // bit 0 of general flags is 0
                   PosDecode,
                   MinFixed,             // Range is fixed
                   MaxFixed,             // Range is Fixed
                   NonCacheable,
                   ReadWrite,
                   0x00000000,           // Granularity
                   0x2000000000,         // Min
                   0x2000FFFFFFF         // Max
                   0x00000000,           // Translation
                   0x0010000000,         // Range Length
                   ,                     // ResourceSourceIndex
                   ,                     // ResourceSource
                   PERB                  // DescriptorName
                )
```

```
                // Interrupt on PMU 0 overflow, GSIV = <gsiv0>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv0>}

                // ..

                // Interrupt on PMU 31 overflow, GSIV = <gsiv31>
                Interrupt (ResourceConsumer, Level, ActiveHigh, Exclusive) {<
                    ↪gsiv31>}
        })
}
```